Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

On the Design and Study of Adaptive Capability for Neural Network Filters in Image Restoration

Pao-Ta Yu and Hung-Hsu Tsai

Department of Computer Science and Information Engineering
National Chung Cheng University
Chiayi, Taiwan 62107, R.O.C.
E-mail: csipty@ccunix.ccu.edu.tw, Fax: 886-5-272-0859

Abstract

The design of adaptive filters becomes more important in the field of image restoration. In this paper, we propose a new class of neural network filters which possess more adaptive capability in contrast to that of rank conditioned rank selection (RCRS) filters which are a large class of selection filters. As we know, the RCRS filters possess some powerful filtering capability among the conventional filters for image restoration. The selection filters include the RCRS filters whose output is one of the observation samples. In addition, the RCRS filters need the exponential computation complexity to achieve their design process. Fortunately, the neural network filters can avoid this problem. Finally, the experiment results demonstrate that the adaptive capability for the neural network filters is better than that of the RCRS filters.

Keywords: neural network, *RCRS* filter, impulsive noise, multilayer perceptrons, back-propagation.

1 Introduction

The adaptive capability of filters becomes more important in the field of image restoration. In practical environment, we may have only a portion of noisy images received from sending station which stores the corresponding original images through the little reliable transmission channels (for instance, the wireless transmission channels) to form some training patterns. That is, we seldom have a complete corrupted images and their corresponding original images to train the neural network filters through the little reliable transmission channels. However, the constructed neural network filters based on the few part of learning messages are still required to perform the filtering restoration with respect to the noisy images which are not recognized by the neural network filters in advance. Henceforth, this reason motivates us to investigate the improved performance of adaptive capability for filtering noises. Many previous works have been toward increasing adaptive capability of noise-removal for the design of their filters[1, 2, 3, 4]. However, the experimental model of those works did not coincide with the practical environments as we mentioned above. In other words, the training patterns are not different from the checking patterns in their experimental models. In contrast, herein another new experimental model we first propose in this paper is that the training patterns are almost different from the checking patterns. Evidently, the experimental model we propose relates closely to the practical situation.

In this paper, we directly employ a multilayer perceptrons ($MLPs$) as the architecture of neural network, and also exert the back-propagation learning algorithm at the same time [5]. The filters designated by this kind of neural networks are called by the NN filters throughout this work. As we know that neural networks are able to approximate nonlinear functions when those input-output measurement patterns (or pairs) are available [6]. Therefore, this concept triggers that the noise-removal problem can be treated as the problem of

system identification [5]. Indeed, our goal is to estimate the behaviors of the system with the given input-output measurement patterns performed by a fixed structure neural network and supervised learning method. Thus, we can say that a set of the given input-output measurement patterns is identified as a set of patterns includes both the training patterns and checking patterns if we are based on the viewpoint of neural network. Meanwhile, it is demanded to achieve the minimization of a certain cost function expressed as a function of weight vector which represents the free parameters of the neural network. The cost function is defined in terms of the derivations of the network outputs from the desired outputs under some criteria, such as the least mean square (LMS) criterion. As we know, the $RCRS$ filters proposed by Hardie *et al.* which are the optimal design filters among the class of ranking selection filters, and are exploited to compare the performance of adaptive capability with the NN filters [7]. In contrast, the NN filters belong to the class of non-selection filters. The experimental results evidently demonstrate that the adaptive capability for the neural network filters is better than that of the $RCRS$ filters.

This paper is organized as follows. Section 2 provides a description of some notations and basic concepts which concern the neural networks and the $RCRS$ filters. The architecture of the NN filters and the learning algorithm applied to them are illustrated in Section 3. The experimental results of comparison between the NN filters and the $RCRS$ filters on several distinct images are shown in Section 4. Finally, we give a simplified conclusion with regard to the adaptive capability of the neural network filters and some future works are also mentioned.

2 Basic Concepts

In this section, we first give some definitions and notations with respect to the image and sliding window. In addition, an appending strategy used widely is also introduced. Next, the definitions of training and checking pattern are stated. Then, the concepts and definitions about the desired functions and the estimated functions implemented by neural networks are illustrated in turn. Finally, we alternately make a brief descriptions for the selection and non-selection filters. For instance, the $RCRS$ filters pertain to the former, whereas the NN filters belong to the latter.

Definition 2.1 *A digital intensity image $\mathcal{I}$ with size $n \times n$ is represented by a matrix defined as follows.*

$$M = \begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ x_{21} & \cdots & x_{2j} & \cdots & x_{2n} \\ \vdots & & \vdots & & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & & \vdots & & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{nn} \end{bmatrix} = [x_{ij}]_{n\times n} \quad (1)$$

$x_{ij} \in \{0,1,2,\cdots,255\}$ is called the grey level of the pixel located at (i,j) in $\mathcal{I}$. Note that $M \in \{0,1,2,\cdots,255\}^{n\times n}$

Definition 2.2 *A sliding window with size $r \times r$ covers on the image $\mathcal{I}$ to obtain an observation vector X_{ij} at position (i,j), and is defined below*

$$X_{ij} = \begin{bmatrix} x_{i-k,,j-k} & \cdots & x_{i-k,j} & \cdots & x_{i-k,j+k} \\ \vdots & & \vdots & & \vdots \\ x_{i,j-k} & \cdots & x_{ij} & \cdots & x_{i,j+k} \\ \vdots & & \vdots & & \vdots \\ x_{i+k,j-k} & \cdots & x_{i-k,j} & \cdots & x_{i+k,j+k} \end{bmatrix} \quad (2)$$

where $k = \lfloor \frac{r}{2} \rfloor$ and $\lfloor \cdot \rfloor$ stands for floor function.

Definition 2.3 *A pattern $P_{ij} = (X_{ij}, d_{ij})$ means that X_{ij} is obtained from a noisy image M_{noisy} and d_{ij} is an observation sample of the original image $M_{original}$ at position (i,j) where $M_{noisy} = [x_{ij}]_{n\times n}$ and $M_{original} = [d_{ij}]_{n\times n}$, $d_{ij} \in \{0,1,2,\cdots,255\}$.*

Example 2.1 *Suppose we already have two images* $M_{original} = \begin{bmatrix} 46 & 46 & 80 & 53 & 76 \\ 51 & 51 & 82 & 50 & 98 \\ 113 & 113 & 67 & 70 & 60 \\ 120 & 120 & 88 & 66 & 70 \\ 58 & 58 & 143 & 61 & 64 \end{bmatrix}$ *and* $M_{noisy} = \begin{bmatrix} 46 & 146 & 0 & 53 & 0 \\ 0 & 51 & 82 & 50 & 98 \\ 113 & 113 & 167 & 70 & 0 \\ 120 & 120 & 88 & 66 & 70 \\ 58 & 58 & 143 & 61 & 164 \end{bmatrix}$, *and also have*

a sliding window with size 3×3 *(i.e.,* $r = 3$ *). A pattern* $P_{33} = (X_{33}, d_{33})$ *is represented by* $X_{33} = \begin{bmatrix} 51 & 82 & 50 \\ 113 & 167 & 70 \\ 120 & 88 & 66 \end{bmatrix} = (51, 82, 50, 113, 167, 70, 120, 88, 6)$ *and* $d_{33} = 67$ *Another pattern* $P_{35} = (X_{35}, d_{35})$ *is represented by* $X_{35} = \begin{bmatrix} 50 & 98 & 98 \\ 70 & 0 & 0 \\ 66 & 70 & 70 \end{bmatrix} = (50, 98, 98, 70, 0, 0, 66, 70, 70)$ *and* $d_{35} = 60$.

Note that one of the appending strategies must be exploited for avoiding the situation that the central position of the sliding window covers those pixels which are near the four boundaries of the image. In this paper, we use the first strategy shown in [8]. In short, it appends the four boundaries pixels several repeated times. The number of repeated times depends on the size of the sliding window to the image formed the appended image. Thus, the given images are applied this kind of appending strategy and are redrawn below. $M_{original}^{appended} = \begin{bmatrix} 46 & 46 & 46 & 80 & 53 & 76 & 76 \\ 46 & 46 & 46 & 80 & 53 & 76 & 76 \\ 51 & 51 & 51 & 82 & 50 & 98 & 98 \\ 113 & 113 & 113 & 67 & 70 & 60 & 60 \\ 120 & 120 & 120 & 88 & 66 & 70 & 70 \\ 58 & 58 & 58 & 143 & 61 & 64 & 64 \\ 64 & 58 & 58 & 143 & 61 & 64 & 64 \end{bmatrix}$ and $M_{noisy} = \begin{bmatrix} 46 & 146 & 0 & 53 & 0 \\ 0 & 51 & 82 & 50 & 98 \\ 113 & 113 & 67 & 70 & 0 \\ 120 & 120 & 88 & 66 & 70 \\ 58 & 58 & 143 & 61 & 164 \end{bmatrix}$ Those pixels located at the top row, the bottom row, the leftmost column and rightmost column of $M_{original}^{appened}$ are appended by means of taking the pixels located at the corresponding boundary, respectively. This example has at most 25 training patterns, i.e., the set of training patterns $T = \{(X_{ij}, d_{ij}) : 1 \le i \le 5, 1 \le j \le 5\}$. Therefore, the cardinality of T is 25.

Subsequently, we know the neural networks can approximate the unknown functions if their input-output measurement patterns are available. We are able to say that the problem of that finding the noise-removal function when both the original and corrupted images are available is akin to the problem of that finding the input-output mapping. If two given images are available, one is an original image represented by $M_{original} = [d_{ij}]_{n \times n}$ and the other is a corrupted version of the original image represented by $M_{noisy} = [x_{ij}]_{n \times n}$. Then, a set of measurement patterns S is specified as follows.

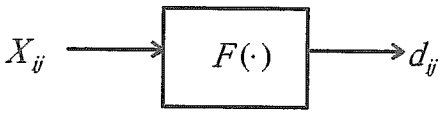


Figure 1: The desired function F.

Definition 2.4 *Suppose that both* $M_{noisy} = [x_{ij}]_{n \times n}$ *and* $M_{original} = [d_{ij}]_{n \times n}$ *are given, and the size of the sliding window is* $r \times r$. *The element* P_{ij} *of* S *is comprised of two components, one is the observation vector* X_{ij} *and the other is the grey level* d_{ij} *of* $M_{original}$ *at position* (i, j). *That is,* $S = \cup \{P_{ij} = (X_{ij}, d_{ij}) : 1 \le i, j \le n\}$ *where* i, j, r *and* n *are positive integers. In general,* $S = T \cup C$ *where* T *is called the set of training patterns, and* C *is called the set of checking patterns and is the difference between* S *and* T.

However, the exact formula of input-output mapping is hard to find it by means of using the approach of mathematics. It is pictorially shown as Fig. 1.

The function $F(\cdot)$ represents the exact formula with respect to the given set of input-output observations patterns, S. In our case, F is defined as

$$F : \{0, 1, 2, \cdots, 255\}^{r \times r} \rightarrow \{0, 1, 2, \cdots, 255\}$$

or an another expression,

$$F(X_{ij}) = d_{ij}.$$

As before, we know that it is difficult to obtain F. Our goal therefore is triggered to approximate F by virtue of neural networks which act as the function $\widehat{F}$ depicted in Fig. 2 such that there are the minimization difference between its actual output $\widehat{d_{ij}}$ and desired output d_{ij}.

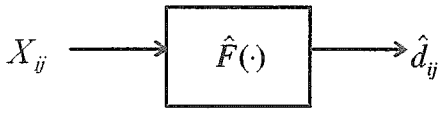


Figure 2: The estimated function $\widehat{F}$.

In summary, a set of input-output observation patterns are transformed to the function performed by a neural network such that this network learns enough about the past to generalize to the future.

Finally, we know the *RCRS* filters which are a large class of selection filters. The output of selection filters is one element of the observation vector fed into the filters (i.e., $d_{ij} \in X_{ij}$). As contrast with the output of the non-selection filters may be or may be not the element of the observation vector (i.e., $\widehat{d_{ij}} \notin X_{ij}$, perhaps). Henceforth, we claim that the NN filters possess more adaptive capability than that of the *RCRS* filters. Fortunately, the experimental results demonstrate the claim we propose is correct.

3 Neural Network Filters

In this section we first introduce the architecture of the NN filters, and then derive the learning algorithm under the least mean square error criterion by virtue of the steepest decent method. Actually, this learning algorithm derived is based on back-propagation algorithm [5]. The difference between our learning algorithm and one of learning algorithms, shown in [5], is the selection of the activation function used by each neuron. This kind of the activation function is exploited by our algorithm to incur two advantages, one is the speed of convergence is faster than the others, and the other is the overflow problem. The latter does not occur during the computing process if the asymmetric activation function is employed.

3.1 Architecture of Neural Network Filters

A p-q-1 multilayer perceptrons is employed to the design of the NN filters, its architectural layout shown in Fig. 3. It needs p input observation samples, such as $x_{i1}, x_{i2}, ..., x_{ip}$, has only one hidden layer with q neurons, and has single output neuron. In Fig. 3, the square stands for one of the input observation samples, and the circle represents the neuron. The neuron model is pictorially depicted in Fig. 4. Those neurons shown in Fig. 3 form a

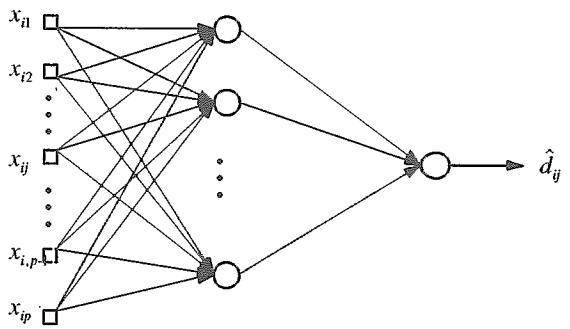


Figure 3: Multillayer perceptron with a single hidden layer and output layer.

fully connected graph which means that each neuron is connected by either all inputs or all neurons located at previous layer. Each connection edge between two neurons has a corresponding weight value. To be specific, each neuron must be given a model depicted as Fig. 4. The neural model in-

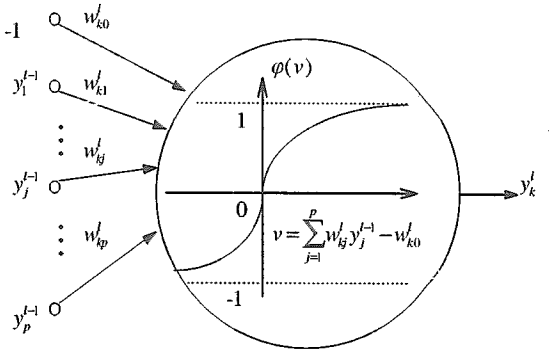


Figure 4: Neuron model of neuron k at layer l.

cludes three basic terms. The first term is a set of synaptic weights, such as $w_{k0}^l, w_{k1}^l, ..., w_{kp}^l$. The second term is a linear combiner,

$$v_k^l = \sum_{j=1}^{p} w_{kj}^l y_j^{l-1} - w_{k0}^l \qquad (3)$$

where w_{k0}^l represents the threshold value. The third term is an activation function φ. In addition, the output of the linear combiner is called internal value, v_k^l, and becomes the input of φ. Then the output of φ is the actual output of the neuron. The details about the model of neuron refer to [5]. As mentioned before, the function of neural network is decided by the weights among connected neurons. In our case, Fig. 2 is also expressed as

$$\widehat{F}(X_{ij}, W) = \widehat{d}_{ij}$$

where W is a set of synaptic weight vectors among connected neurons. Next, it is important that what sort of the learning algorithms is able to compute W to a specific structure neural network such that it can achieve the minimum difference between all of the desired outputs and the corresponding actual outputs.

3.2 Learning Algorithm

In fact, if the neural network wants to work well, it then must include not only a fixed structure $MLPs$, but also it needs a learning algorithm to decide W. In analogy to the well-known back-propagation learning algorithm, our learning algorithm is derived where the activation function is different from [5]. Our activation function is the use of a hyperbolic tangent function, a sigmoidal nonlinear function, which is asymmetric corresponding to origin and for its amplitude of output between -1 and 1, shown in Fig. 4. Our learning algorithm adopts a hyperbolic tangent function, φ, to be the activation function in which φ is defined as

$$\varphi(v) = a\tanh(bv) = a\left[\frac{1-e^{(-bv)}}{1+e^{(-bv)}}\right] \qquad (4)$$

where the parameters a and b denote the amplitude of $\varphi's$ outputs and $\varphi's$ slope, respectively. The shape of φ is depicted in Fig. 4 when parameter a is equal to 1.

The learning algorithm is actually called as the updated rule of the weights shown as Eq. (5), and is derived when Eq. (4) is chosen as the activation function of a neuron.

$$w_{ji}^l(n+1) = w_{ji}^l(n) + \eta\delta_j^l(n)y_i^{l-1}(n) \qquad (5)$$

The term $w_{ji}^l(n)$ means that the weight between neuron i at the $(l-1)$th layer and neuron j at the lth layer, and is obtained after the nth training pattern had been learned. η is a learning rate which is a small constant. $y_i^{l-1}(n)$ (i.e., $\varphi(v_j^{l-1})$ is derived by Eq. (3) and Eq. (4)) represents the actual output of neuron i at the $(l-1)$th layer after the nth training pattern had been learned, and is computed during the forward-propagation phase. δ_j^l stands for the local gradient of neuron j at the lth layer, and is computed during the backward propagation phase. However, computing δ_j^l must consider two cases, either when neuron j is an output neuron or an hidden neuron. The corresponding two formulas are represented by Eq. (6) and Eq. (7), respectively. The formula of the former case is

$$\begin{aligned}\delta_j^L(n) &= e_j^L(n)\varphi_j'(v_j^L(n)) \qquad (6)\\ &= e_j^L(n)\left[\frac{b^2\left(a^2-(o_j(n))^2\right)}{2a}\right]\end{aligned}$$

where $\delta_j^L(n)$ is the local gradient for neuron j at the output layer L, $o_j(n)$ is the actual output of the neuron j (i.e., $o_j(n) = y_j^L(n)$), and $e_j^L(n)$ is the difference between the desired output $d_j(n)$ and $o_j(n)$, expressed as $e_j^L(n) = d_j(n) - o_j(n)$. Then, the latter case has

$$\begin{aligned}\delta_j^l(n) &= \varphi_j'(v_j^l(n))\sum_k \delta_k^{l+1}w_{kj}^{l+1} \qquad (7)\\ &= \left[\frac{b^2\left(a^2-\left(y_j^l(n)\right)^2\right)}{2a}\right]\left(\sum_k \delta_k^{l+1}w_{kj}^{l+1}\right)\end{aligned}$$

where $\delta_j^l(n)$ is the local gradient for neuron j at the hidden layer l.

4 Experimental Results

Two methods of the construction for a set of measurement patterns S which includes both the set of training patterns T and checking patterns C, and which is obtained from an original and a corrupted version of the image are introduced. Then, some experimental results which concern the performance of the adaptive capability for the $RCRS$ and the NN filters are exposed.

I^1	I^2	I^3	I^4
I^5	I^6	I^7	I^8
I^9	I^{10}	I^{11}	I^{12}
I^{13}	I^{14}	I^{15}	I^{16}

Table 1: Partitioned image

In the experiment, a 2-dimensional image $\mathcal{I}$, basically, can be partitioned into 16 subimages, each subimage has the same length and width, depicted as Table 1, where $\mathcal{I}=\{I^i : i = 1, \cdots, 16\}$, I^i denotes a subimage of $\mathcal{I}$ and can be represented by Eq. (1). Suppose we have the original (desired) image $\mathcal{I}_{original}=\left\{I^i_{original} : i = 1, \cdots, 16\right\}$ and a corrupted version of the image $\mathcal{I}_{noisy}=\{I^i_{noisy} : i = 1, \cdots, 16\}$. Both $\mathcal{I}_{original}$ and $\mathcal{I}_{noisy}$ are formed as Table 1 and represented by Eq. (1), as $M_{original} = [x_{ij}]_{n\times n}$ and $M_{noisy} = [d_{ij}]_{n\times n}$, respectively. If a sliding window is available, then a set of measurement patterns $S_{\mathcal{I}}$ which includes both the set of training patterns $T_{\mathcal{I}}$ and checking patterns $C_{\mathcal{I}}$ is set up in which

$$T_{\mathcal{I}} = \{P^s : \text{the } s\text{th subimage } I^s \text{ in } \mathcal{I} \text{ is selected to be trained}, s \in \{1, 2, \cdots, 16\}\} \quad (8)$$

and

$$C_{\mathcal{I}} = \{P^s : \text{the } s\text{th subimage } I^s \text{ in } \mathcal{I} \text{ is selected to be checked}, s \in \{1, 2, \cdots, 16\}\} \quad (9)$$

where

$$P^s = \{(X_{i,j}, d_{ij}) : X_{ij} \text{ is obtained when the sliding window masks } I^s_{noisy} \text{ at position } (i, j) \text{ and } d_{ij} \text{ is the grey level of the pixel located at } (i, j) \text{ in } I^s_{original} \text{ where } I^s_{noisy} \text{ denotes the } s\text{th subimage in } \mathcal{I}_{noisy} \text{ and } I^s_{original} \text{ represents the } s\text{th subimage in } \mathcal{I}_{original}\}.$$

Three images, "Lenna", "Bridge", and "Baboon", with size 512×512 are employed in the experiment. Their noisy images are their corresponding original images corrupted by impulse noise which is ubiquitous in the transmission channel such as the satellite communication channel. The sort of noises is so-called salt-pepper noise which includes positive and negative impulse [9]. The noise rate of those noisy images exploited in this experiment is 10% positive impulse and negative impulse (i.e., 10% impulsive noise). From Definition 2.4 and Eq.s (8) - (9), we have Eq. (10) when a pair of two images, an original image of "Lenna" and a corrupted version of "Lenna", are available. In short, to let $\mathcal{I}$ be image "Lenna".

$$S^1_{Lenna} = T^1_{Lenna} \cup C^1_{Lenna} \quad (10)$$

where S^1_{Lenna} denotes a set of measurement patterns,

$$T^1_{Lenna} = \{P^s \mid I^s, s = 2k - 1 \text{ and } 1 \le k \le 8\}$$

and

$$C^1_{Lenna} = \{P^s \mid I^s, s = 2k \text{ and } 1 \le k \le 8\}$$

Suppose the method used to construct the set S^1_{Lenna} is called the first method, for convenient explanation. We introduce the second method to generate another set of measurement patterns S^2_{Lenna} when we make an exchange of the elements of T^1_{Lenna} and C^1_{Lenna}. Therefore, S^2_{Lenna} is expressed by Eq. (11).

$$S^2_{Lenna} = T^2_{Lenna} \cup C^2_{Lenna} \quad (11)$$

where

$$T^2_{Lenna} = \{P^s \mid I^s, s = 2k \text{ and } 1 \le k \le 8\}$$

and

$$C^2_{Lenna} = \{P^s \mid I^s, s = 2k + 1 \text{ and } 1 \le k \le 8\}.$$

In analogy to the construction of S^1_{Lenna} and S^2_{Lenna} for image "Lenna", both image "Bridge" and image "Baboon" have the sets of measurement patterns, S^1_{bridge}, S^2_{bridge}, S^1_{baboon} and S^2_{baboon}, respectively. In the experiment, a $RCRS$ filter with order 1 and a NN filter whose architecture is 25-9-1 $MLPs$ are employed. The experimental results are shown in Table 2 and 3 when a 5×5 sliding window is available to generate those sets of measurement patterns using the first and second method, respectively. We show the results of image "Bridge" in Fig 5. Only those subimages which are exploited to generate the checking patterns are restored in each image. Therefore, the claim we propose in Section 2 is verified by the experimental results.

	S^1_{Lenna}	S^1_{bridge}	S^1_{baboon}
$RCRS$ filter	26.37	151.78	223.31
NN filter	17.17	78.94	96.18

Table 2: Comparion of simulated results for a $RCRS$ filter with order 1 and a NN filter based on mean square error (MSE) criterion when the set of measurement patterns is generated by the first method.

	S^2_{Lenna}	S^2_{bridge}	S^2_{baboon}
$RCRS$ filter	27.44	162.91	206.53
NN filter	18.75	90.17	84.28

Table 3: Comparion of simlulated results for a $RCRS$ filter with order 1 and a NN filter based on mean square error (MSE) criterion when the set of measurement patterns is generated by the second method.

5 Conclusions

In this paper, a new approach of experiment related to study the performance of adaptive capability for the design of filters has been proposed. The experiment results demonstrate that the $RCRS$ filters possess less adaptive capability than that of the NN filters. In addition, one of the problems is that the $RCRS$ filters need the exponential computation complexity to achieve their filtering process. The NN filters can avoid this problem, but there are three factors that affect the requirement of the number of computational operations to the NN filters. The first factor is the size of the set of training patterns, the second factor is the number of neurons, and the last factor is the number of epochs. The NN filters are constructed by a fixed $MLPs$ structure and are applied by a supervised learning method. Thus, the design of the NN filters also needs a large number of computational operations when the number or size mentioned in the three factors increases. Therefore, we know the adaptive capability performed well by the design of neural network filters. Unfortunately, a large number of computational operations is required in the training phase. Thus, how to reduce the time of the training phase and how to improve the performance of adaptive capability at the same time are the most important problems.

References

[1] P.-T. Yu and R.-C. Chen, "Fuzzy Stack Filters - Their Definitions, Fundamental Properties, and Application in Image Processing," *IEEE Trans. on Image Processing*, vol. 5, no. 6, pp.838-854, June 1996.

[2] R.-C. Chen and P.-T. Yu, "An Optimal Design of Fuzzy (m, n) Rank Order Filtering with Hard Decision Neural Learning," *Journal of Information Science and Engineering*, 11, pp. 567-593, 1995.

[3] L. Yin, J. Astola, and Y. Neuvo, "A New Class of Nonlinear Filters - Neural Filters," *IEEE Trans. on Signal Processing*, vol. 41, no. 3, pp. 1201-1222, March 1993.

[4] L. Yin, J. Astola, and Y. Neuvo, "Adaptive Neural Filter," *First IEEE-SP Work shop on Neural Networks for Signal processing*, Sept. 29-Oct. 2, 1991, Princeton, New Jersey, USA, pp. 503-512.

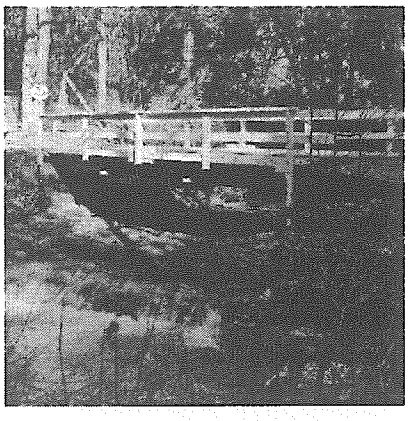
[5] S. Haykin, *Neural Networks*, Macmillan College Publishing Company, 1995.

[6] M. F. Tenorio, and W. T. Lee, "Self-Organizing network for Optimum Supervised Learning," *IEEE Trans. on Neural Networks*, vol. 1, No. 1, March 1990.

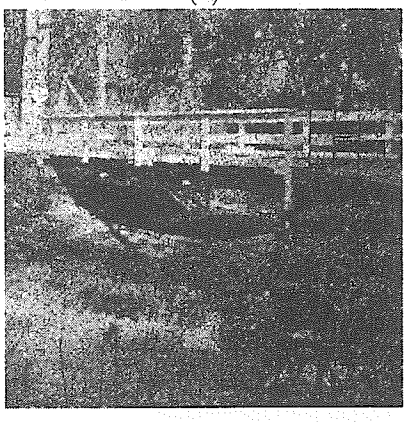
[7] R. C. Hardie and K. E. Barner, "Rank Conditioned Rank Selection Filters for Signal Restoration," *IEEE Trans. on Image Processing*, vol. 3, no. 2, pp.192-206, March 1994.

[8] P.-T. Yu and W. -L. Wang, "Root properties of median filters Under Three Appending Strategies," *IEEE Trans. on Signal Processing*, vol. 41, no. 2, February 1993.

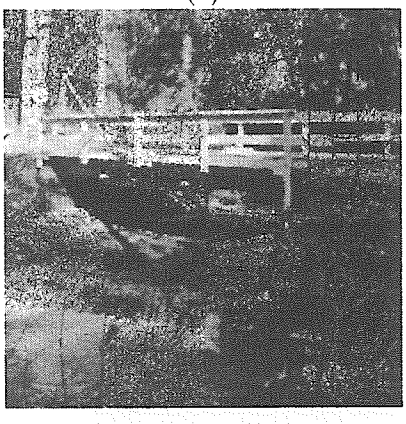
[9] I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters - Principles and Applications*, Kluwer Academic Publishers, 1990.



(a)

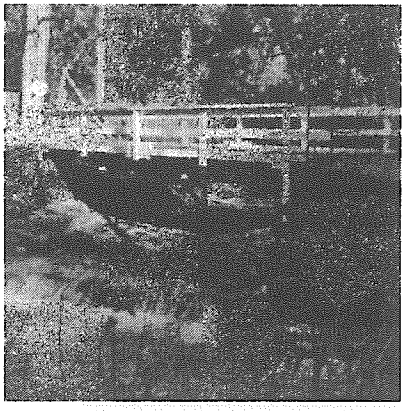


(b)

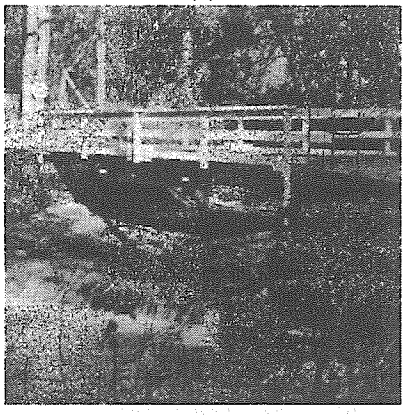


(c)

(d)



(e)



(f)

Fig. 5. (a) Original image "Bridge", (b) Original image "Bridge" corrupted by 10% impulsive noise, (c) Impulse corrupted image "Bridge" restored using a 5×5 *RCRS* filter with order 1 when the measurement patterns are generated by the first method, (d) Impulse corrupted image "Bridge" restored using a *NN* filter with a 5×5 sliding window when the measurement patterns are generated by the first method, (e) Impulse corrupted image "Bridge" restored using a 5×5 *RCRS* filter with order 1 when the measurement patterns are generated by the second method, and (f) Impulse corrupted image "Bridge" restored using a *NN* filter with a 5×5 sliding window when the measurement patterns are generated by the second method.