# COMPARATIVE STUDY ON THE GENERALIZED ADAPTIVE NEURAL FILTER WITH OTHER NONLINEAR FILTERS

*Henry Hanek, Nirwan Ansari and Zeeman Z. Zhang*

Center for Communications and Signal Processing
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
University Heights
Newark, New Jersey 07102
USA

## ABSTRACT

The Generalized Adaptive Neural Filter (*GANF*) is a new type of adaptable filter. The GANF relies upon neural functions to set up a filtering operation. This paper looks at a few of the possible neural operators which can be used in a GANF. The capabilities of the neural nets are examined and the filtering abilities of the GANF are obtained through simulation. While the GANF structure used here is somewhat simplified, the filter is also compared to other non-adaptive filters. These filters provide a reference so that relative performance can be more realistically judged.

## 1. INTRODUCTION

There are situations encountered in signal processing where nonlinear filtering is required. Included in these are the recovery of signals corrupted by non-Gaussian noise [1]. One class of filters, called stack filters, are known to suppress non-AWG (Additive White Gaussian) noise well. Stack filters [1] are defined by the two properties which they possess: the threshold decomposition property and the stacking property. These properties allow the stack filter to break a filtering operation down into a group of parallel binary operations. However, just as with other nonlinear filters, optimal stack filters often cannot be practically designed. As a result, a few methods of adaptive stack filtering have been proposed [2] -[4]. Recently, a new type of adaptive filter, based on the stack architecture has emerged. The Generalized Adaptive Neural Filter (*GANF*), developed by Ansari *et al* [5], uses artificial neural elements in place of the stack filter's Boolean function. Figure 1 shows a stack filter. Here, an input signal of integers from the set {0,1,..,M} is filtered. To determine each output sample, the filter processes only a finite window of input samples, denoted by $\underline{r}$. Let $r(n)$ denote the $n^{th}$ element of $r(n)$. The stack structure dictates that the filter consists of M-1 levels. For each of these levels, the windowed section of the input signal is threshold decomposed according to $T_{(n)}^{level} = \begin{cases} 1 & if \ r(n) \geq level \\ 0 & if \ r(n) < level \end{cases}$. This has the effect of producing a "stack" of 0's on top of a "stack" of 1's for each of the windowed integers. Adding a complete column produces the decomposed integer. In a stack filter, there are M-1 identical binary filters, each processing the binary information on their own level to produce M-1 bits of output. These bits are added to yield the filter output for a given index. It should be noted that identical, positive Boolean functions on each level produce output bits which always stack (i.e.- 1's never above 0's in the column).

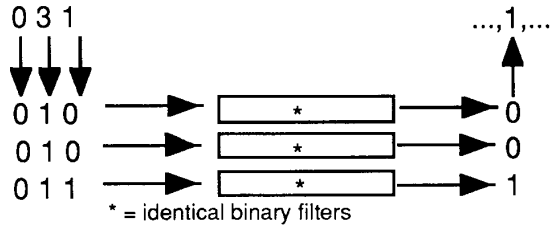



Figure 1. Stack Filter

Figure 2 shows a section of the GANF. Its structure is borrowed from that of the stack filter, but the filtering has been modified in two important ways. First, the identical fixed binary operators of the stack filter are replaced by independent neural operators. Second, the GANF structure allows the neurons on individual levels to receive inputs from adjacent levels on the stack. These changes allow the filter to be adaptive and have more capabilities, but the stacking property may no longer be preserved. (With different Boolean functions implemented on each of the levels, the binary outputs may not necessarily stack anymore.) As discussed in [5], the GANF has been shown to be quite effective in the suppression of non-AWG noise.

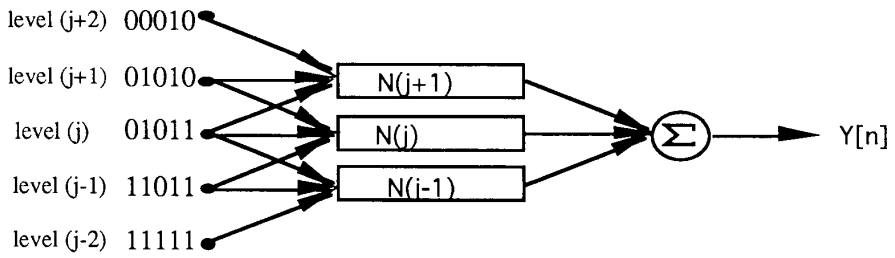



Figure 2. A section of the GANF.



0-7803-0946-4/93 $3.00 

## 2. NEURAL OPERATORS

The neural operators on each level of the GANF perform a classification of the binary input data. Certain binary patterns produce a level output of "1," while others produce a "0" output. The classification operation of each neural element determines the overall filtering operation of the GANF. These neural elements are, of course, configured through an adaptive training procedure. There are many choices of neural operators and training schemes. We will consider a few possibilities.

In the simplest case, a single neuron can be used to implement a Boolean function. This scheme was used for stack filter adaptation in [2]. However, a single neuron cannot implement all possible Boolean functions for a given input vector length. For example, let b equal the window size of the GANF. This will result in a total input vector length of N=b(2I+1), where I equals the number of adjacent levels fed in. There are then $2^{2^N}$ possible Boolean functions or classifications which the neuron could be called upon to perform. However, only a fraction of these can be realized by a single neuron [6]. As a result, if complicated classifications are required, such a neural operator will limit the capabilities of the GANF.

To develop a neural network capable of implementing any Boolean function, first consider the case where there are two inputs (N=2). Two binary inputs specifies four possible input combinations. Most of the class assignments for these four possibilities can be realized with a single neuron. The only difficulty arises when XOR functions must be implemented. A solution to this involves using an additional neuron. Two hyperplanes can be used to separate both of the product terms in the XOR Boolean function. The outputs of each neuron can then be ORed together (by a third neuron) to produce the final class assignment. It can be easily proved that the third neuron can perform the OR operation. Therefore, a two layer, three neuron network can implement any Boolean function in 2 variables. Now, if we want to classify an input vector of length 3 or more, the number of possible classifications get squared. Adding an input to a network with N-1 inputs yields two independent classifications of the N-1 inputs – one set of class assignments for the Nth input high and another for the Nth input low. So the solution can be implemented by doubling the structure that already exists. Each independent structure can be ANDed with $x_N$ and $\overline{x}_N$.

*PROPOSITION*: A single neuron which implements a particular Boolean function can be made to implement the ANDing of this function with a particular state of an added (Nth) input. That is, for one state of $x_N$ the neuron output will always be low, while the other state of $x_N$ will allow a certain classification of the other N-1 inputs. (proof omitted due to lack of space.)

Using this idea, Figure 3 shows the structure needed to classify a 3 bit input. Generally, a two layer net with $2^{N-1}$ neurons on the first layer and 1 neuron in the second layer can implement any Boolean function of N variables.

Another possibility for implementing any Boolean function is to pre-process the input data in a fixed manner. One way of doing this can be seen by considering a one input case. A linear discriminant function (LDF) capable of arbitrarily classifying this input is:

$$g_1(x_1) = w_0 + w_1 x_1$$

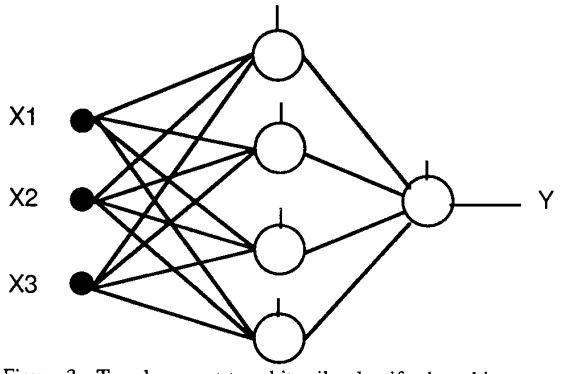



Figure 3. Two layer net to arbitrarily classify three binary inputs

Another classification can be represented by

$$g_2(x_1) = w_A + w_B x_1$$

Now, combining these as

$$g(\underline{x}) = (1 - x_2) g_1(x_1) + x_2 g_2(x_1)$$

any classification in 2 variables can be achieved. Here, we are letting $\underline{x}$ represent the input vector $(x_1, x_2)$. $\underline{x}$ is classified according to $g_1(x_1)$ if $x_2 = 0$ and by $g_2(x_1)$ if $x_2 = 1$. Simplifying yields the expression,

$$g(\underline{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2$$

where $w_2 = w_A - w_0$ and $w_3 = w_B - w_1$.

Each time a new input is added, the new LDF has the form of the old LDF plus the old LDF times the new input. From this, it can be seen that a single neuron with $2^N$ pre-processing functions can implement any Boolean function of N variables.

A third possibility is to use a 3 layer network [7][8], which can implement any function defined on $[0,1]^N$. Such a network would have N first layer neurons, (2N+1) second layer neurons and 1 output neuron.

Finally, a compromise between the single neuron and the more complicated arrangements can be examined. One possibility is to use pre-processing in a limited sense. That is, the input vector can be processed by something such as a quadric function. Then the neural classification abilities would be better than that of a single perceptron, but less than that of the other 3 networks discussed.

## 3. EXPERIMENTAL RESULTS AND CONCLUSIONS

For simulation, the clean image shown in Figure 4a was corrupted with an ε-mixture of Gaussian noise to create the image in Figure 4b. Four separate GANFs were set up to filter the image. Each of these GANFs used one of the neural networks previously discussed. The 3 layer and Quadric networks were used in GANF structures with both 3x3 and 5x5 window sizes. Due to their complexity, the 2 layer and complete pre-processing networks were simulated only for a 3x3 window. All the GANFs used independent neural functions on each of the 255 levels and no adjacent level

information was used. Backpropagation was used to train the 2 and 3 layer nets, while the LMS rule was used for the two pre-processing nets. The filters were trained on the lower right hand corner of the image. Then the weights were fixed and the remaining section of the image was filtered.

In addition, the noisy image was filtered with conventional, non-adaptive nonlinear filters. All of the filtering results are shown in Table 1, where only the upper left hand three-quarters of the images were considered. The non-adaptive filters were taken from [9]. However, to clarify the Wilcoxon, ver. 1 allowed the indexes j and l (as shown in [9]) to be equal while ver. 2 did not.

The GANFs performed rather well, both in terms of error and in subjective impression. Figure 4c shows the image after filtering by the quadric GANF, while Figure 4d shows it for the 2 layer GANF. Figures 4e and 4f show the best results for the non-adaptive filters. Theoretically, the 2 layer, complete pre-processing and 3 layer nets all had the same capabilities, which were greater than that of the quadric. However, the network with the least separation abilities (the quadric) produced the best results. This suggests that for the more complicated networks, the training algorithms did not allow convergence to a minimum error solution. While complex networks can handle more precise pattern classification, they must be trained properly to take advantage of this.

In conclusion, the effectiveness of the GANF was demonstrated. However, it was also shown that the structure of the neural operators need not be overly complex to achieve good results. Complicated networks should do better in terms of error, but they really slow down the operation of the filter. In addition, inadequate learning can easily cancel the benefits of complete separation. GANF structure may therefore be dependent on such things as window size, amount of training data and expected usage. Future work could involve improvements in training schemes along with GANF structure improvements. These improvements will be necessary to increase performance both in terms of error and filtering speed.

## 4. REFERENCES


[1] P. D. Wendt, E. J. Coyle and N. C. Gallagher, "Stack filters," *IEEE Trans. ASSP*, vol. ASSP-34, pp. 898-911, Aug. 1986.

[2] N. Ansari, Y. Huang and J. Lin, "Configuring stack filters by the LMS algorithms," *First IEEE-SP workshop on neural networks for signal processing*, Sept. 29 - Oct. 21, 1991, Princeton, NJ, pp. 570-579.

[3] B. Zeng, H. Zhou and Y. Neuvo, "FIR Stack hybrid filters," *Optical Engineering*, vol. 30, pp. 965-975, July 1991.

[4] J.H. Lin, T.M. Sellke and E.J. Coyle, "Adaptive Stack Filtering Under the Mean Absolute Error Criterion," *IEEE Trans. ASSP*, vol. 38, no. 6, pp. 938-954, June 1990.

[5] Z.Z. Zhang, N. Ansari and J. Lin, "On Generalized Adaptive Neural Filters," *Proc. IJCNN'92*, June 7-11, 1992, Baltimore, MD, pp. IV.277-282.

[6] N.J. Nilsson, *"The Mathematical Foundations of Learning Machines,"* Morgan Kaufmann Publishers, CA, 1990.

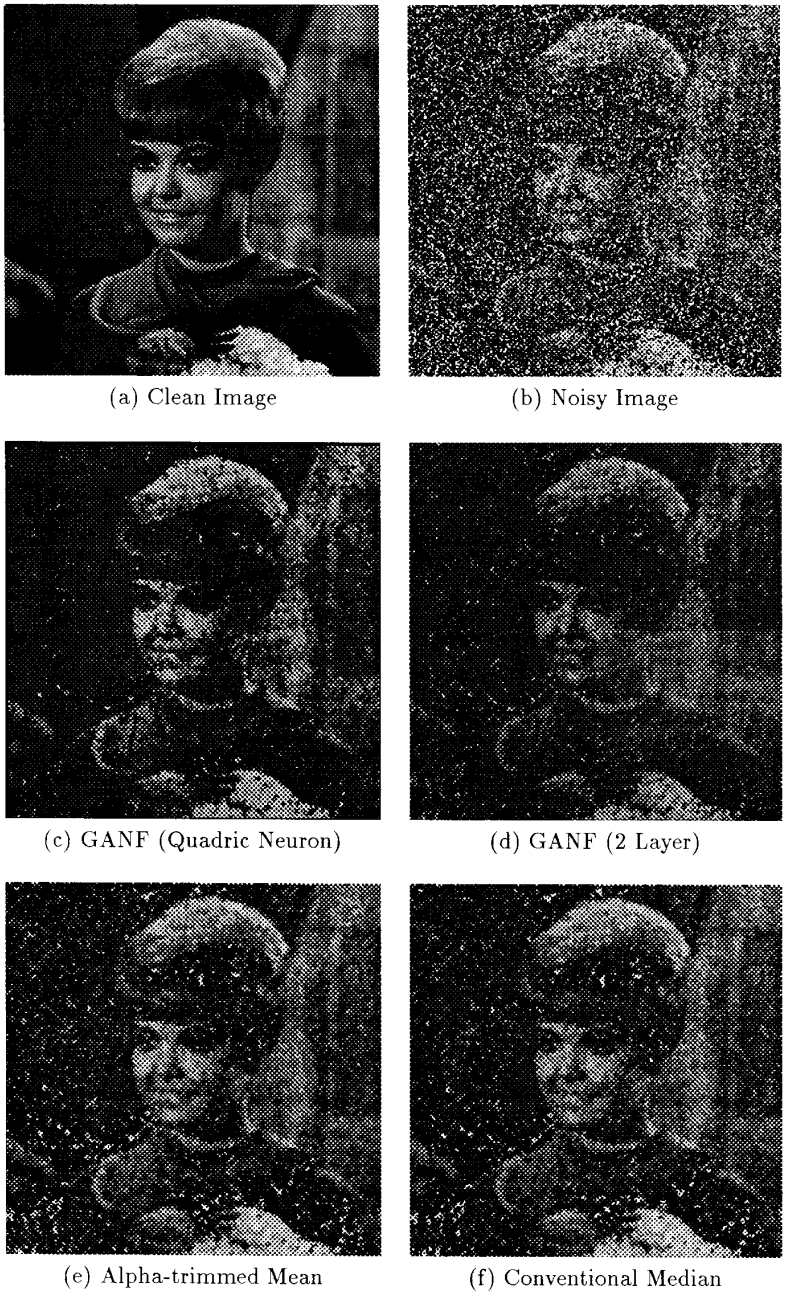
[7] R. Hecht-Nielson, "Kolmogorov's Mapping Neural Network Existence Theorem," *Proc. IEEE Internat. Conf. of Neural Networks,* June 21-24, 1987, San Diego, CA, pp. III.11-13.

[8] R. Hecht-Nielson, "Theory of the Backpropagation Neural Network," *Internat. Joint Conf. on Neural Networks,* June 18-22, 1989,Washington, DC, Vol. 1, pp. 593-605.

[9] Y.S. Fong, C.A. Pomalaza-Raez, X.H. Wang, "Comparison study of nonlinear filters in image processing applications," *Optical Engineering*, vol. 29 no. 7, July 1989.


| FILTER | 3X3 WINDOW SIZE | | | 5X5 WINDOW SIZE | | |
|---|---|---|---|---|---|---|
| | MAE | MSE | SNR [dB] | MAE | MSE | SNR [dB] |
| Clean Image | 0 | 0 | ∞ | 0 | 0 | ∞ |
| Noisy Image | 44.46 | 5067.89 | 0.98 | 44.46 | 5067.89 | 0.98 |
| GANF (2 layer) | 20.56 | 742.60 | 9.32 | --- | --- | --- |
| GANF (Complete pre-processing) | 22.89 | 1104.38 | 7.60 | --- | --- | --- |
| GANF (3 layer) | 30.55 | 1477.29 | 6.33 | 15.63 | 480.86 | 11.21 |
| GANF (Quadric pre-processing) | 15.74 | 602.51 | 10.23 | 13.04 | 427.31 | 11.72 |
| Alpha-trimmed mean, alpha=0.4 | 19.64 | 932.70 | 8.33 | 14.97 | 464.17 | 11.36 |
| Modified trimmed mean, q=10 | 18.58 | 946.74 | 8.26 | 14.30 | 456.28 | 11.44 |
| K-nearest neighbor v.1, K=6 (3x3), 18 (5x5) | 25.89 | 1653.16 | 5.84 | 21.39 | 1082.42 | 7.68 |
| K-nearest neighbor v. 2, K=4, 12 | 25.20 | 1707.14 | 5.70 | 24.59 | 1584.61 | 6.03 |
| Modified K-nearest neighbor, v.1, K=3, 9 | 18.04 | 952.70 | 8.24 | 12.88 | 382.30 | 12.20 |
| Modified K-nearest neighbor, v.2, K=2, 6 | 18.27 | 1006.61 | 8.00 | 13.36 | 410.88 | 11.89 |
| Wilcoxon v.1 | 25.30 | 1364.70 | 6.68 | 22.09 | 966.60 | 8.18 |
| Wilcoxon v.2 | 27.19 | 1524.23 | 6.20 | 23.61 | 1092.05 | 7.64 |
| Adaptive Mean, C=100 | 33.24 | 1524.23 | 6.20 | 23.61 | 1092.05 | 7.64 |
| Adaptive Median, C=100 | 31.02 | 3864.55 | 2.16 | 27.62 | 3578.59 | 2.49 |
| Conventional Median | 18.51 | 929.47 | 8.34 | 14.50 | 450.70 | 11.49 |
| Separate Median | 20.02 | 1106.39 | 7.59 | 15.45 | 530.88 | 10.78 |
| Max/median | 52.87 | 6013.65 | 0.24 | 45.31 | 4283.38 | 1.71 |

Table 1.



(a) Clean Image (b) Noisy Image

(c) GANF (Quadric Neuron) (d) GANF (2 Layer)

(e) Alpha-trimmed Mean (f) Conventional Median

Figure 4: Experimental results with $3 \times 3$ window.