Fast Training of the Adaptive Neural Stack Filter

Mounir Sayadi*, Farhat Fnaiech* and Daniel Bastard**

*ESSTT, University of Tunis, 5 Av. Taha Hussein, 1008, Tunis, Tunisia,
Fax +2161391166, Email : sayadi@goelette.tsi.u-bordeaux.fr

**Equipe Signal & Image, ENSERB Av. Albert Schweitzer, BP 99, F33402, Talence cedex, France

Abstract:

In this paper, a fast approach for training the adaptive neural stack filter based on the Recursive Least Square algorithm is presented. The architecture of the neural stack filter is transformed in order to allow the fast recursive estimation of the adaptive neural filter weights. This approach is compared to the Least Mean Square training algorithm (also called Back-propagation algorithm in case of multilayer neural network). Computer simulations on restoration of noisy images are presented. It is shown that the proposed fast training algorithm is 2.5 time faster than the LMS algorithm.

1. Introduction:

The stack filters, introduced in [5], form a large class of easily implemented nonlinear filters, they share the threshold decomposition and the stacking properties of rank-order filter. A *M*-level (1D or 2D) signal is mapped into *M*-1 binary signals by thresholding the original signal at each of the *M* levels. The set of *M*-1 signals is then filtered by *M*-1 Boolean operators having stacking properties. The multilevel output is finally obtained as the sum of the *M*-1 binary output signals. The adaptive neural stack filter is a non linear adaptive filter introduced in [6] to overcome the shortcomings of linear filters and to decrease the complexity of the classical stack filter by using a neural network [2] to represent the Boolean functions of the stack filter. For the training of the neural filter, Yin et al [6] derive the adaptive least mean absolute and the adaptive least mean square training algorithms and compare their performances with other widely used filters.

In this paper, we propose a fast training approach for the adaptive neural stack filter based on the Recursive Least Square algorithm. This training approach is introduced in [1] and [4] for the multilayer neural network. A stabilized version of this method based on U-D factorization is also proposed in [3] for the multilayer neural network.

2. Training of the adaptive neural stack filter:

The role of the adaptive neural stack filter is to remove the additive impulsive noise from a given image of *M* colors. The process is applied on a window $\underline{R}$ of ($n\times n$) pixels. A training phase, where the filter weights are adjusted, is then needed. The adaptive neural stack filter training scheme is given in Figure 1, where R, $\hat{S}$ and S denote the corrupted, filtered and original images, respectively.

3. Threshold decomposition and neural stack filter [6]:

Let us consider the window $\underline{R}$ ($n\times n$ pixels) of the input image ($N_{\max}\times N_{\max}$ pixels) given in the following matrix form

$$\underline{R}=\begin{bmatrix} \underline{R}_{11} & \dots & \underline{R}_{1n} \\ .. & \dots & .. \\ \underline{R}_{n1} & \dots & \underline{R}_{nn} \end{bmatrix}$$

where $\underline{R}_{ij}$ denotes the color of i,j pixel.

Let us define the matrices $\underline{r}^m$ by :

$$\underline{r}^m = \begin{bmatrix} \underline{r}_{11}^m & \cdots & \underline{r}_{1n}^m \\ . & \cdots & . \\ \underline{r}_{n1}^m & \cdots & \underline{r}_{nn}^m \end{bmatrix}$$

as the threshold decomposition of this window where m is the threshold of the decomposition (m : 1 to M-1) and

$$\underline{r}_{ij}^m = \begin{cases} 1 & \text{if } R_{ij} \geq m \\ 0 & \text{otherwise} \end{cases}.$$

By stacking 2.I+1 threshold vectors where I is an integer, we obtain for each threshold m the matrix $r^m = \left[\underline{r}^{m+I}\, \underline{r}^{m+I-1} \ldots\ldots\ldots \underline{r}^{m-I}\right]^t$.

The matrices r^m can be considered as the input of a single layer neural network, called neural stack filter, in which the weights are concatenated into the following matrix :

$$w^m = \left[\underline{w}^{m+I}\, \underline{w}^{m+I-1} \ldots . \underline{w}^{m-I}\right]^t$$

$$\text{where } \underline{w}^m = \begin{bmatrix} w_{11}^m & \cdots & w_{1n}^m \\ . & \cdots & . \\ w_{n1}^m & \cdots & w_{nn}^m \end{bmatrix}.$$

With this matrix form of both the neural filter inputs and the weights, it is difficult to use an algorithm different from the classical LMS training algorithm presented in [6].

4. New adaptive neural filter architecture:

We propose to modify the filter architecture presented in [6] by transforming each input matrix r^m into an input vector x^m of (d+1) elements (d=n^2.(2.I+1)) given by:

$$x^m = \left[x_0^m \ldots\ldots\ldots\ x_d^m\right] =$$

$$\left[1\ r_{11}^m\, r_{12}^m \ldots .\ r_{1n}^m\, r_{21}^m\, r_{22}^m \ldots\ldots\ldots\ldots r_{nn}^m\right].$$

The adaptive filter weights are then concatenated into a corresponding vector. Hence we can use the recursive least square approach, used in [1] and [4] for the multilayer feedforward neural networks, to train the adaptive neural stack filter.

The output of each neuron is noted by :

$\hat{S}^m$ (where m: 1... M-1).

The summation of these outputs gives $\hat{S}_{ij}$ (Figure 2) which is the estimation of S_{ij} representing the central pixel color of the desired image window. Let S^m denotes the decomposition of S_{ij} corresponding to the threshold m : (i,j: 1.. $N_{\max}$).

$$S^m = \begin{cases} 1 & \text{if } S_{ij} \geq m \\ 0 & \text{otherwise} \end{cases}$$

The different steps of the RLS algorithm for the training of the adaptive neural stack filterare given by:

1-Initialize the weights w_i^m with arbitrary values.

2-For each training iteration t, calculate the filter output: $\hat{S}^m(t) = f\left(y^m(t)\right)$ using

$$y^m(t) = \sum_{i=0}^{d} x_i^m(t).w_i^m(t)$$

Where f is a sigmoïd function given by
$f(y)$= 1/(1+exp(-a.y))
and a the sigmoïd slope.

3-Compute the Kalman gain

$$k_m(t) = \frac{R_m^{-1}(t).x_m(t)}{b + x_m^T(t).R_m^{-1}(t).x_m(t)}$$

and adjust the inversion matrix $R_m^{-1}(t)$ using:
$R_m^{-1}(t) = \left(R_m^{-1}(t) - k_m(t).x_m^T(t).R_m^{-1}(t)\right)/\lambda$
with λ a forgetting factor. $k_m(t)$ is a (d+1) vector and $R_m^{-1}(t)$ is a (d+1)×(d+1) matrix.

4-Compute the desired linear output:

$$d^m(t) = -1/a.Log\left(\frac{1-S^m(t)}{S^m(t)}\right).$$

5-Adjust the filter weights using:

$$w_i^m(t) = w_i^m(t-1) + \mu\left(d^m(t) - y^m(t)\right)k_i^m(t)$$

where μ is the step-size parameter.

6-Verify the stack filter constraint:

if $w_i^{m+1}(t) > w_i^m(t)$ then $w_i^m(t) = w_i^{m+1}(t)$.

5. Computer simulation:

The LMS and the RLS algorithms are used for training an adaptive neural stack filter to restore a noisy image, of 128×128 pixels, corrupted by additive impulsive noise. We use *M*=256; *n*=3 and *I*=0. The impulses probability is 0.1. The sigmoïd slope is 0.1 and the initial random values of the weights are between -5 and 5. The step-size parameter of the LMS training algorithm are 0.3. For the RLS training algorithm, the step size parameter is 0.3 and the forgetting factor is 0.999. The mean square error (MSE) is defined as

$$E = \frac{1}{N_{\max}^2}\sum_{i=1}^{N_{\max}}\sum_{j=1}^{N_{\max}}(\hat{S}_{ij} - S_{ij})^2 .$$

Note that one training iteration corresponds to the filtering of the quarter of the training image.
The training performances of the two algorithms are compared in table 1. The training is carried out with the quarter of the image and the convergence threshold is chosen to be 0.1. Figures 4, 5 and 6 show the original, corrupted and filterd images, respectively, with the neural stack filter trained by the RLS algorithm.

Algorithms	LMS	RLS
convergence iterations	21	6
convergence time(s)	1711	675
iteration duration (s)	81	112

Performances of the two algorithms.
(Training with the quarter of the image)

6. Conclusion :

In this paper, a fast training approach for the adaptive neural stack filter based on the RLS algorithm is presented. We transform the architecture of the neural stack filter to allow the fast recursive estimation of the adaptive neural filter weights. This approach is compared to the LMS training algorithm. Computer simulations on the restoration of noisy images are presented. They show that the proposed fast training algorithm is 2.5 time faster than the LMS algorithm.

References:

[1] M.R. Azimi-Sadjadi and R.J. Liou," Fast Learning Process of Multilayer Neural Networks Using Recursive Least Squares Method," *IEEE Transactions on Signal Processing,* vol.40, no. 2. pp.246-450, February 1992.

[2] R.P. Lippmann," An introduction to computing with Neural Networks," *IEEE ASSP Magazine,* vol.4, no.2, pp 4-22, April 1987.

[3] M. Sayadi, F. Fnaiech, A. Chaari et M. Najim, "Apprentissage rapide des réseaux de neurones multicouches : Application de l'algorithme des moindres carrées récursifs factorisés," *Proc. International Conference " Les réseaux neuromimétiques et leurs applications,* pp. 311-322, 15-16 December 1994, Marseille, (France).

[4] R. S. Scalero and N. Tepedelenlioglu, "A Fast New Algorithm for Training Feedforward Neural Networks," *IEEE Transactions on Signal Processing,* vol.SP-40,. no. 1, pp. 202-210, January 1992.

[5] P. D. Wendt, E. J. Coyle and N. C. Gallagher JR, "Stack Filters," *IEEE Transactions on Acoustics Speech and Signal Processing,* vol. ASSP-34, no. 4, pp. 898-911, August 1986.

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

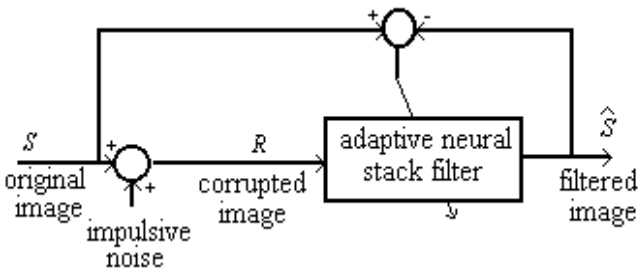

Figure 1: Adaptive neural stack filter training scheme

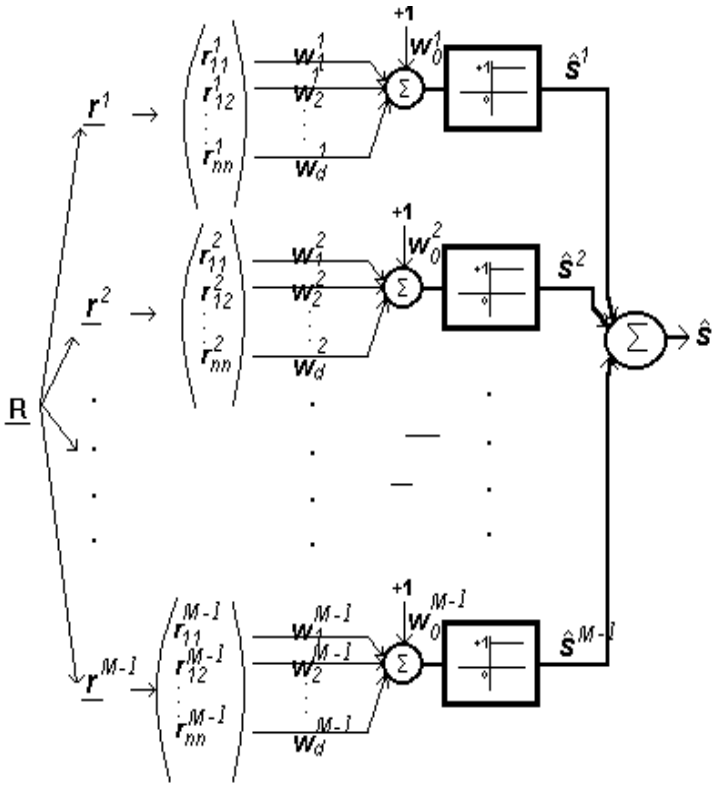

Figure 2: New adaptive neural filter architecture

Figure 3: Neural stack filter scheme

Figure 4 : Original image

Figure 5 : Corrupted image (MSE=87.21)

Figure 6 : Filtered image with the neural stack filter trained by the RLS algorithm (MSE=12.36)