Neural Network Combined with Fuzzy Logic to Remove Salt and Pepper Noise in Digital Images

A. Faro, D. Giordano, C. Spampinato
Dipartimento di Ingegneria e Informatica e delle telecomunicazioni
Università degli studi di Catania, viale A. Doria 6, Catania, 95125 Italy

Abstract. Image denoising is an important step in the pre-processing of images. Aim of the paper is to remove the salt and pepper noise on images by using a novel filter based on neural network and fuzzy logic. By this filter it is possible to remove only the pixels that are really affected by noise, thus avoiding image distortion due to the removal of good pixels. A comparison between the proposed filter and the classical median filter shows an increase of about 20% of the peak signal to noise ratio and a better capacity of preserving the details of the images. The proposed approach outperforms the existing algorithms and does not depend on the noise level.

1 Introduction

Digital images are affected by different kinds of noise such as Gaussian noise, random noise with normal distribution, salt and pepper noise, and black and white impulses on the digital image. Gaussian noise is usually the result of a camera electronic noise, whereas salt and pepper noise is the result of inoperative or 'dead' pixel within the camera sensor. Linear filters are most frequently used to suppress the noise because they may be easily designed and implemented. However, linear filters are suitable to eliminate linear noises. For other kinds of noise, the performance of such filters highly decreases and specific denoising methods have to be adopted.

This paper deals with how to manage the salt and pepper noise, that is an impulsive noise with uniform spatial distribution in the image where each noisy dot can be either an isolated pixel or composed of more than one pixel. Median filters [1] have been widely used for removing such impulsive noise, since they are quite effective for the noise removal and easy to implement. However, a median filter assigns to each pixel P the median value of the pixels belonging to a small window around P, and therefore the filter removes image details, being little sensible to the extreme values of the scale (e.g., the

black and white of the grey tone scale), and also causes image distortion because it is applied to all the pixels of the image. The former problem has been widely investigated and may be solved by assigning to any pixel the value computed by an algorithm, let us say algorithm I, that aims at minimizing a suitable objective function [2]. During the minimization procedure the luminosity of most of the uncorrupted pixels will converge to the uncorrupted values whereas the other pixels will be restored preserving the edges and the local features. Since not all the uncorrupted pixels remain unchanged, algorithm I may cause out of focus and distorted images too. For this reason other algorithms, let us say algorithms II, have been proposed (e.g., [3]) to identify the noisy pixels so that the previous algorithm I is applied only to restore the corrupted pixels. In [4] another algorithm, let us say algorithm III, is proposed that shows better performances. It consists in the application of the algorithm II followed by the algorithm I in which the objective function takes into account only the uncorrupted pixels around the corrupted pixel to be restored. However, some performance limitations of the proposed algorithms I, II and III have been pointed out too. These limitations deal with the high processing time for image denoising and the rapid decrease of performance at increasing levels of the salt and pepper noise.

Aim of the paper is to show that by a soft computing approach it is possible to restore the images corrupted by salt and pepper noise with performances better than the ones obtainable by the available algorithms. The proposed approach makes use of neural networks to identify the set S of the pixels that are potentially affected by the salt and pepper noise and of fuzzy logic to avoid considering affected by noise pixels that belong to S but that have not been corrupted. The restoration of the noisy pixels is carried out by the median filter that takes into account only the uncorrupted pixels around the corrupted pixel to be restored. The use of objective functions, such as the one proposed in [2], instead of the median filter for better preserving the edges and the local features of the images is for further study. The paper is organized as follows: Sect. 2 addresses the use of neural network to understand what are the potentially noisy pixels. Section. 3 focuses on the use of fuzzy logic to choose the threshold for denoising the image. Section. 4 describes the results of processing real images and finally Sect. 5 reports the conclusions.

2 Neural Network Approach for Noisy Pixel Identification

Multi-layer perceptron networks trained by back propagation are among the most popular and versatile form of neural networks. The input vector representing the pattern to recognize is passed to the input layer and distributed to the subsequent hidden layer and to the output layer via weighted connections.

Each neuron in the network operates by taking the sum of its weighted inputs and passing the result through a nonlinear activation function.

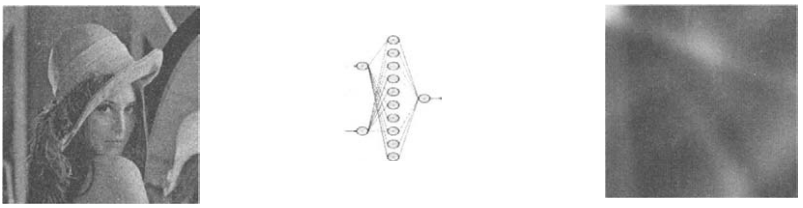
Multi-layer neural networks are well known for their capability of approximating non linear bi-dimensional functions such as the one that represents the luminosity $L(x,y)$ of an image whose pixels (x,y) are possibly corrupted by a salt and pepper noise. In our approach the original image is approximated by an image extracted from the original one by using a neural network having two input neurons related to the coordinates (x,y), some intermediate neurons and an output neuron related to the luminosity L_a of the pixel of coordinates (x,y), *i.e.*, $L_a(x,y)$. The network tries during the training phase of reproducing $L(x,y)$. However, due to the non-linearity of the function $L(x,y)$ it is not possible for the neural network to reproduce exactly this function. The best that can done is to modify its synaptic weights by a generalized delta rule that aims at minimizing the distance between $L_a(x,y)$ and $L(x,y)$.

Interestingly, due to the high variations of the luminosity of the pixels affected by the salt and pepper noise with respect to the neighboring pixels, the distance between the luminosity of the pixels corrupted by the noise and the one learned during the testing phase remains very high since the neural network is not able to follow the high luminosity variations produced by the noise. This feature allows us to use the image learnt by the neural network as a filter, being possible to isolate the potentially noisy pixels as the ones of the learnt image that are characterized by a high distance from the corresponding pixels belonging to the original image.

To illustrate how the method works, we show in Fig. 1a the original image I whose pixel luminosity is given by the luminosity $L(x,y)$ affected by the salt and pepper noise, whereas Fig.1c points out the approximated image I_A whose pixel luminosity is given by the function $L_a(x,y)$ extracted from image I by using the neural network drawn in Fig.1.b. The hidden layer consists of ten neurons which have proved adequate for our scope. By subtracting the approximated image from the original one we obtain the error image I_E, drawn in Fig. 2a, whose pixel luminosity is given by $\Delta L(x,y) = L(x,y) - L_a(x,y)$. Let $P_N(x,y)$ be the pixels that are potentially covered by noise, to identify them we have to decide a threshold T so that such pixels are the ones for which the following relation holds :

$$\Delta L\,(x,y) > T$$

The final result of the method consists of a mainly black image in which the noisy pixels $P_N(x,y)$ are pointed out by white points (Fig. 2b). By applying the outlined median filter to only the pixels $P_N(x,y)$ we would have a better performance, in terms of signal to noise ratio, than the one reachable with the conventional median filter.



(a) (b) (c)

Fig. 1. (a) Corrupted image, (b) Neural Network used, (c) Image approximated by the neural network

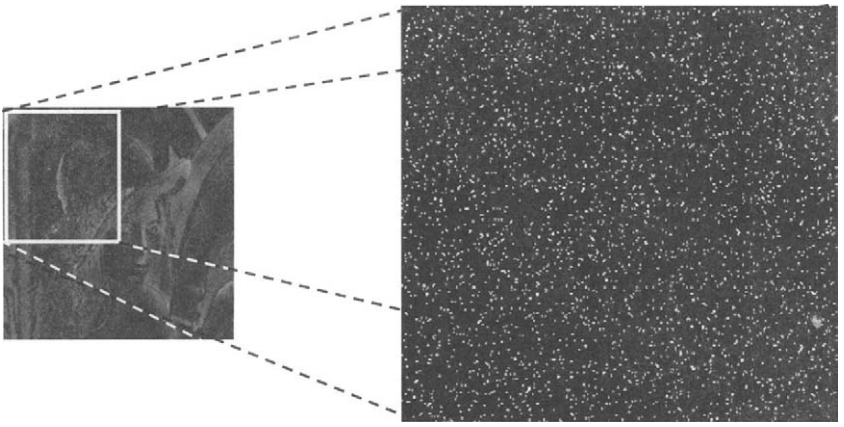


Fig. 2. (a) Error image, (b) Thresholded image related to a part of the error image

Since the value of the threshold T may influence significantly the performance of the proposed method, such decision is taken empirically by taking into account the mean value μ and the variance σ of the pixel luminosity, i.e.,:

$$\mu = \Sigma_{i,j} L(xi,yi) / N \qquad \sigma = (\Sigma_{i,j} (L(xi,yi) - \mu)^2 / N)^{1/2}$$

where N is the total number of the image pixels. In the next section we will propose a method to choose such threshold by a fuzzy logic approach, thus increasing the performance of the neural networks based filter.

3 Fuzzy Thresholding

To find the rules that may help in finding the optimal threshold for extract-ing the noisy pixels from a set of pixels potentially affected by noise, the Mamdani fuzzy controller may be used [5]. In fact, such a system works very well

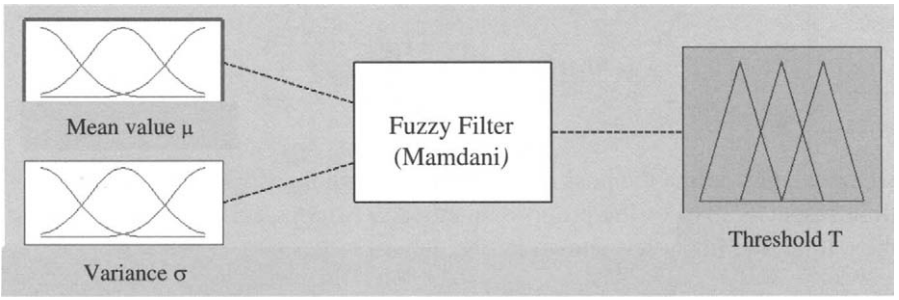


Fig. 3. Mamdani System to threshold error image

for identifying the fuzzy rules able to interrelate a set of known input-output pairs. In our case, the fuzzy controller (see Fig. 3) will receive, as inputs, the previously defined average value μ and the variance σ of the error image pixel luminosities, whereas the output variable has to be the value T to threshold the error image.

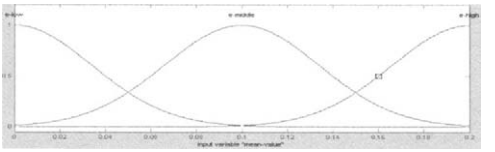
However, whereas $\mu(i)$ and $\sigma(i)$ may be easily computed from the images, to find the threshold T(i) we should proceed according to the following procedure:

- take a set S of different images I(i) characterized by different mean luminosities $\mu(i)$ and variances $\sigma(i)$
- add a salt and pepper noise to the previous images thus obtaining the images $I_N(i)$
- apply the method outlined in Sect. 2 thus obtaining from images $I_N(i)$ the image M(i) consisting, as the one drawn in Fig. 2.b, of a wide black area containing some white pixels representing the pixels potentially affected by the noise
- find the threshold T(i) that allow us for each image $I_N(i)$ to identify more or less all the noisy pixels

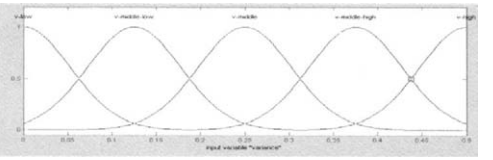
The above {input-output} pairs $\{\mu(i), \sigma(i) - T(i)\}$, are useful to train the Mamdani fuzzy controller in order to automatically find the set of fuzzy rules able to identify the unknown threshold for any image that has not been processed during the training phase. In order to implement the fuzzy thresholding, the following two-parameters Gaussian type membership function is used:

$$m(u) = \exp\left(-\left(\left(\frac{u-c}{s}\right)^2\right)\right)$$

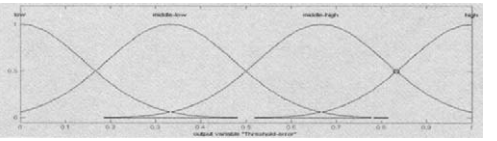
where c and s define the position of the centre and the slope of Gaussian function, respectively. For the proposed method, we adopt, for the two inputs and the output, the fuzzy sets shown in Fig. 4.



mean value μ
low, middle, high



variance σ
low, middle-low, middle, middle-high, high



threshold T
low, middle-low, middle high, high

Fig. 4 Fuzzy sets for the mean value, the variance, and the threshold.

After having trained the Mamdani fuzzy controller with a set of training images, the following rules to threshold the error image have been obtained:
IF (μ is low) AND (σ is low) THEN (threshold is low)
IF (μ is low) AND (σ is middle-low) THEN (threshold is low)
IF (μ is low) AND (σ is middle-high) THEN (threshold is middle-high)
IF (μ is middle) AND (σ is middle) THEN (threshold is middle-low)
IF (μ is middle) AND (σ is middle-high) THEN (threshold is middle-high)
IF (μ is middle) AND (σ is high) THEN (threshold is high)

IF (μ is high) AND (σ is middle-low) THEN (threshold is middle-high)
IF (μ is high) AND (variance is middle-high) THEN (threshold is high)

Figure 5 points out how these rules allows us to compute the threshold starting from mean-value and variance. It is easy for the user to slightly modify these rules or to add new ones depending on her/his experience. This could be done, for example, modifying the previous rules by using adjectives such as *quite*, *most*, *essentially* and *little*, e.g.: *IF (mean-value is low or quite medium) AND (variance is essentially low) THEN (threshold is little low).* To compute the membership functions of the modified rules, the interested reader may consult [6].

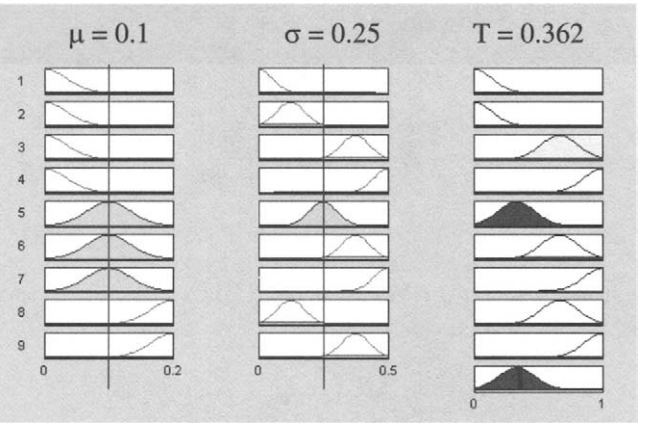


Fig. 5 Fuzzy rules dealing with mean value, variance, and the output.

To speed up the process of finding the threshold it is suitable to compute in advance the function T(μ(i), σ(i)). Figure 6 shows the function we have obtained according to the outlined procedure.

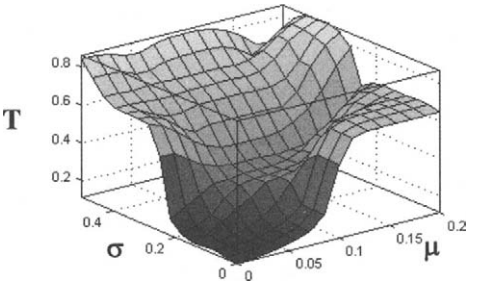


Fig. 6. Fuzzy control surface

4 Results of the Image Pre-Processing

The noise dealt with our algorithms is only the salt and pepper impulse noise, where the corrupted pixels take on the value of either 0 or 255 with equal probability. The percentage of noisy pixels R is used to indicate how much an image is corrupted. For example, if an image is corrupted by R=10% impulse noise, then 10% of the pixels in the image are corrupted randomly by positive and negative impulses. The algorithm has been tested on the Lena Image shown in Fig. 7.

Fig. 7. Original Lena image

Peak signal-to-noise ratio (PSNR) is used to evaluate the restoration performance. PSNR is defined as:

$$PSNR = 10\log_{10}\left\{\frac{255^2}{\frac{1}{MN}\sum(x_{i,j}-y_{i,j})^2}\right\}$$

where $x_{i,j}$ e $y_{i,j}$ are the coordinates of the pixels P(i,j) of the source image and the restored image respectively. The image size is MxN and 8 bit/pixel. Figure 8 shows the results using the median filter, the iterative (10 iterations) median filter which, as is known, increases the performance of the median filter and the proposed filter, for an image corrupted with salt and pepper noise with R=10%. The simple median filter can preserve the image details but many noisy pixels remain in the image. The iterative median filter removes most of the impulses, but many good pixels are also removed. The proposed filter removes almost all the pixels that are really affected by noise.

The PSNR performances provided in Table 1 show that by our approach we obtain a significant increase of performances and that the decrease of PSRN is little influenced by the increase of the noise level. The advantage of using our approach is particularly evident if one considers the performances of a sophisticated algorithm of type III, such as the one presented in [4]. In fact, in this algorithm PSRN(30% of noise) = 36.5 and PSRN(50% of noise) = 33, whereas in our algorithm PSNR is about 38.3 and 37.2 respectively.

Fig. 8. Corrupted Lena image (a), and outputs of: median filter with 3x3 window (b), ten iterations of median filter (c), and proposed algorithm (d)

Table 1. PSNR of the restored images for different noise levels by the median filter and by our algorithm. The last column refers to a double iterations of our algorithm.

% Noise	Median Filter	Iterative Median Filter	Proposed Algorithm	Iterative version of the proposed Algorithm
10	35.497	34.551	37.881	38.984
30	32.155	31.811	35.537	38.342
50	30.708	30.456	33.410	37.191

To evaluate the performances of the algorithm we also use the SAD (sum of absolute difference) between the original image and the filtered image. Such parameter gives an idea on how much the details of the original image are preserved. SAD is defined as follows:

$$SAD = \frac{1}{NxM} \sum\sum \left| u(i,j) - y(i,j) \right|$$

where $u_{i,j}$ e $y_{i,j}$ are the coordinates of the pixels P(i,j) of the original image and the restored image respectively. For the image Lena the results shown in Table 2 demonstrate the good performances achieved by our algorithm also for preserving the details of the image. Figure 9 confirms qualitatively this result by showing an image in which edges and local features play a role more relevant than the one they play in the Lena image denoised by our algorithm. PSRN and SAD related to this case show the same figures of the previous ones, thus it could not be necessary to use objective functions (e.g., the one

Table 2. SAD of the restored images for different noise levels by the median filter and our algorithm. The last column refers to a double iterations of our algorithm.

% Noise	Median Filter	Iterative Median Filter	Proposed Algorithm	Iterative version of the proposed Algorithm
10	1.004	1.580	0.407	0.180
30	1.741	1.816	1.289	0.340
50	4.003	2.075	1.975	0.410

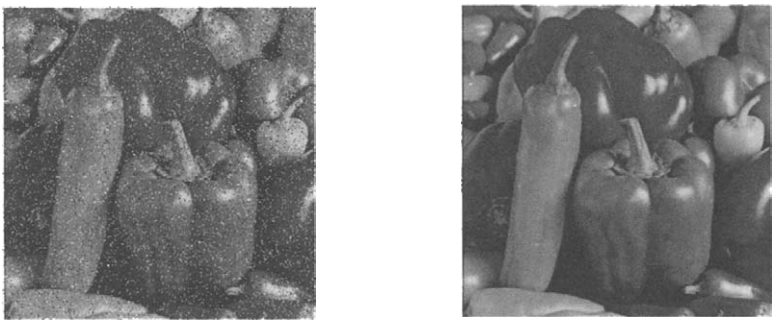


Fig. 9. Corrupted peppers image with 10% noise on the left and output of the iterative proposed algorithm on the right

proposed in [2]) instead of the median filter for preserving the details of the images. In this way we can avoid to significantly increase the processing time being the objective function approaches more time consuming than the ones based on the median filter [4].

A complete discussion on the processing time is for further study. However, let us note that the CPU time needed for processing, by a PC equipped with a 2 GHz CPU, the Lena and the peppers images covered by 50% of noise is about 1000 seconds even if the noise increases to 70–90%. This execution time is less than the time (i.e., 2009 seconds) needed by the algorithm III at 70% of noise (see [4]). Moreover, the execution time needed by our algorithm does not depend on the noise levels neither on the image complexity, whereas the one required by the algorithm III is highly influenced by these factors (e.g. it passes from 2009 to 6917 seconds when the noise increases from 70% to 90%, and from 2009 seconds for denoising the Lena image to 4263 seconds for denoising the image of a bridge). We don't have information about the time needed by the algorithm III at lower levels of noise. Presumably its execution time should decrease. However, it is likely that the execution time of our algorithm will still be better because it is considerably less than the one of algorithm III at 70% of noise, and for lower noise levels it could further

decrease at the expenses of a PSRN lower than the one currently attained by using a lower number of epochs during the learning phase of the neural net.

5 Concluding Remarks

A novel type of filter based on neural networks and fuzzy logic has been proposed and applied to noise reduction in digital images. In this filter, we use the neural network to approximate the function $L(x,y)$ that represents the image luminosity. Next, we use fuzzy logic to threshold the error image, given by the difference between the original image and the approximated one, thus obtaining the pixels that are really affected by the salt and pepper noise. Experiments indicate that our method provides a significant improvement over the median filter. A comparison with other algorithms proposed in literature has shown that our approach outperforms the available algorithms in both PSRN and SAD, thus obtaining images that are very close to the original ones even if the images show relevant edges. The time performance of our approach is better than the one of the existing algorithms at the same noise levels. Its time performance does not depend on the noise level neither on the image complexity, whereas the one of the other algorithms are highly influenced by these factors. Better PSRN and SAD could be achieved by a repetitive use of the proposed filter (see last column of tables 1 and 2). Since this increases the processing time, we are evaluating if and when this is acceptable or if a GRID based implementation may be more suitable for the problem at hand.

References

[1] Gonzales R.C.and Woods R.E., (2002): *Digital image processing*. Prentice Hall

[2] Nikolova, M. (2004): *A variational approach to remove outliers and impulse noise*, Journal of Mathematical Imaging and Vision 20 (2004), pp. 99-120

[3] Hwang H., Haddad, R.A., (1995): *Adaptive median filters: anew algorithms and results*, IEEE Transactions on Image Processing 4 (1995) pp. 499-502

[4] Chan R.H., Ho C.W. Nikolova M. (2005): *Salt and pepper noise removal by median type noise detectors and detail-preserving regularization* – www.math.cuhk.edu.hk/~rchan/paper/impulse/impulse.pdf to appear on IEEE Transactions on Image processing

[5] Mamdani, E. H., Assilian, S. (1975): *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller*. Int. Journal of Man-Machine Studies, 7 (1) pp. 1-13

[6] Wang P.P. (Ed.) (2001): *Computing With Words*. Wiley